Provable Security for Public Key Schemes

Various Models and Methods

Barcelona – February 2004

David Pointcheval
CNRS-ENS, Paris, France

Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
  - The Ideal-Cipher Model
  - The Generic Model
    - LD is hard
    - ECDSA
- Shoup's Proof Technique
Strong Security Notions

- Signature: difficult to obtain security against existential forgeries for CMA
- Encryption: difficult to reach CCA security
- Maybe possible, but with inefficient schemes
- Inefficient schemes are useless in practice:

  Everybody wants security, but only if it is transparent

Ideal Models

→ One makes some ideal assumptions:
  - ideal random hash function:
    - random-oracle model
  - ideal symmetric encryption:
    - ideal-cipher model
  - ideal group:
    - generic model (generic adversaries)
Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
    - The Ideal-Cipher Model
    - The Generic Model
      - LD is hard
      - ECDSA
  - Shoup's Proof Technique

The Random-Oracle Model

Introduced by Bellare-Rogaway  ACM-CCS '93

- The most admitted model
- It consists in considering some functions as perfectly random functions, or replacing them by random oracles:
  - each new query is returned a random answer
  - a same query asked twice receives twice the same answer
Modeling a Random Oracle

A usual way to model a random oracle $H$ is to maintain a list $\Lambda_H$ which contains all the query-answers $(x, \rho)$:

- $\Lambda_H$ is initially set to an empty list
- A query $x$ to $H$ is answered the following way
  - if for some $\rho$, $(x, \rho) \in \Lambda_H$, $\rho$ is returned
  - Otherwise,
    - a random $\rho$ is drawn from the appropriate range
    - $(x, \rho)$ is appended to $\Lambda_H$
    - $\rho$ is returned

Modeling a Random Oracle (Cont'd)

Two equivalent views:

- $H$ is a random function
  - a query $x$ to $H$ is answered by $H(x)$
  - $\rho_1, \rho_2, \ldots$ is a random sequence of answers
  - a new query $x$ to $H$ is answered by the next element in the sequence
Summary

- Ideal Models
  - The Random-Oracle Model
  - The Forking Lemma
  - The Ideal-Cipher Model
  - The Generic Model
    - LD is hard
    - ECDSA
- Shoup's Proof Technique

Schnorr Signature (1989)

- $G, g$ and $q$: common elements
- $x$: private key  $y=g^x$: public key

**Signing $m$:**
- choose $k \in \mathbb{Z}_q$
- compute $r=g^k$ as well as $e=H(m,r)$
- and $s = k - xe \mod q$

**Verifying $(m, \sigma)$:**
- $u = g^s y^e \quad (= g^{k-xe} g^{xe})$
- test if $e = H(m,r)$ and $r = u$
Security Proof  Pointcheval-Stern '96

Existential Forgery = DL problem

- Idea: forking lemma

\[ A \xrightarrow{H(m,r)} e \xrightarrow{e'} (r,e,s) \]

\[ g^s \cdot y^e = r = g^{s'} \cdot y^{e'} \]

Let \( \alpha = (s-s') / (e'-e) \mod q \)
Then \( y = g^\alpha \)

Existential Forgery

- A asks \( q_H \) queries \((m_pr_i)_i\): \( h_i = H(m_pr_i) \) and
  outputs \((m^*, r^*, e^*, s^*)\) such that \( \mathbb{V}(m^*, r^*, e^*, s^*) = 1 \)
  with success probability \( \nu \)

- With probability \( \varepsilon \),
  there is \( i \) such that \( (m^*, r^*) = (m_pr_i) \)

- \( H \) over \( \ell \) bits: \( \varepsilon \geq \nu - 1/2^\ell \)
because
  \[ \Pr[H(m^*, r^*) = e^*] = 1/2^\ell \]

\[ \varepsilon = \Pr[\text{Success}] \]
\[ \varepsilon_i = \Pr[\text{Success} \land m^* = m_i] \]
\[ \varepsilon = \sum \varepsilon_i \]
**Forking Lemma**

- For any $i$, one defines
  \[\Omega = \{(\omega, h_1, \ldots, h_{i-1}, h_i, \ldots, h_{q_H})\} = X_i \times Y_i\]
  \[X_i = \{x_i = (\omega, h_1, \ldots, h_{i-1})\}\]
  \[Y_i = \{y_i = (h_i, \ldots, h_{q_H})\}\]

\[\varepsilon_i = \Pr_{X_i \times Y_i}[\text{Success} \land m^* = m_i] \]

\[Z_i = \left\{ x_i \in X_i \mid \Pr_{Y_i}[\text{Success} \land m^* = m_i] \geq \alpha_i \right\} \]

---

**Splitting Lemma**

- Assume $\Pr[x_i \in Z_i] < \varepsilon_i - \alpha_i$

\[\varepsilon_i = \Pr[S_i] = \Pr[S_i \mid x_i \in Z_i] \Pr[x_i \in Z_i] + \Pr[S_i \mid x_i \notin Z_i] \Pr[x_i \notin Z_i] \]

\[< 1 \times (\varepsilon_i - \alpha_i) + \alpha_i \times 1 = \varepsilon_i \]

\[\Pr[Z_i \mid S_i] = 1 - \Pr[x_i \notin Z_i \mid S_i] = 1 - \Pr[S_i \mid x_i \notin Z_i] \Pr[x_i \notin Z_i] / \Pr[S_i] \geq 1 - \alpha_i \times 1 / \varepsilon_i \]

\[\Pr[x_i \in Z_i \mid S_i] \geq 1 - \alpha_i / \varepsilon_i \]
Forking Lemma - 2

- Run A once: for any $i$
  - success and $m^* = m_i$ with probability greater than $\varepsilon_i$
  - $x_i \in Z_i$ with probability greater than $1 - \alpha_i / \varepsilon_i$
- Run A a second time with same $x_i$ but random $y_i$
  - new success with probability greater than $\alpha_i$

\[
p = \sum \varepsilon_i \times \left(1 - \frac{\alpha_i}{\varepsilon_i}\right) \times \alpha_i = \sum (\varepsilon_i - \alpha_i) \times \alpha_i
\]

Forking Lemma - 3

With $\alpha_i = \rho \varepsilon_i$

\[
p = \sum \varepsilon_i \times \left(1 - \frac{\alpha_i}{\varepsilon_i}\right) \times \alpha_i = \sum (\varepsilon_i - \alpha_i) \times \alpha_i
\]

\[
= \sum \left(\varepsilon_i^2 (1-\rho) \times \rho\right) = (1-\rho)\rho \times \sum \varepsilon_i^2 \\
\geq (1-\rho)\rho \times (\sum \varepsilon_i)^2 / q_H = (1-\rho)\rho \times \varepsilon^2 / q_H
\]

Optimal for $\rho = 1/2 : p \geq \varepsilon^2 / 4 q_H$
Forking Lemma - 4

- Run $A$ once with random $(\omega, h_1, \ldots, h_{i-1}, h_i, \ldots h_\ell) = (x_i, y_i)$
- In case of success:
  - run $A$ again with same $x_i$ but random $y_i'$
- One gets two successes
  - $(m_1, r_1, e_1, s_1)$ and $(m_2, r_2, e_2, s_2)$
  - such that $(m_1, r_1) = (m_2, r_2)$
  - $\mathbb{V}(m_1, r_1, e_1, s_1) = 1$ and $\mathbb{V}(m_2, r_2, e_2, s_2) = 1$
  - with probability greater than $\varepsilon^2 / 4 q_H$

---

Forking Lemma - Details

- From $\varepsilon$ to $\nu$
- Two successes
  - $(m_1, r_1, e_1, s_1)$ and $(m_2, r_2, e_2, s_2)$
  - such that $(m_1, r_1) = (m_2, r_2)$
  - and $e_1 \neq e_2$
- Success with probability greater than
  - $((\nu - 1/2^\ell) \times ((\nu - 1/2^\ell)/4 q_H - 1/2^\ell)$
  - $\nu^2 / 4 q_H - \nu / 2^\ell (1 + 1/2 q_H)$
Forking Lemma - Improvement

- Run A until one gets a success:
  - on average = \(1/\varepsilon\) iterations: for any \(i\)
    - \(m^* = m_i\) with prob greater than \(Pr[S_i | S] \geq \varepsilon / \varepsilon\)
    - \(x_i \in Z_i\) with probability greater than \(1 - \alpha_i / \varepsilon_i\)
- Run A again with same \(x_p\) but random \(y_i\)
  - until a success: on average \(1 / \alpha_i\) times
- On average:

\[
T = \frac{1}{\varepsilon} + \sum \frac{\varepsilon_i}{\varepsilon} \times \left(1 - \frac{\alpha_i}{\varepsilon_i}\right) \times \frac{1}{\alpha_i} = \frac{1}{\varepsilon} + \frac{1}{\varepsilon} \times \sum \frac{1 - \rho}{\rho} = q_H + \frac{1}{\varepsilon}
\]

---

Chosen-Message Attacks

- The random oracle provides an easy simulation of the signing oracle (ZK).
- The forking lemma applies to:
  - Fiat-Shamir
  - Guillou-Quisquater
  - Schnorr
  - …
- Generalization to any identification scheme secure against passive adversaries [BP02]
Forking Lemma - Comments

- Security bound: $2^{75}$ and $2^{55}$ hash queries
- If one can break the scheme within time $T = t/\epsilon$, one can extract two tuples within time $T' \leq q_H t/\epsilon = q_H T \leq 2^{130}$
- RSA (GQ Signature)
  - 1024 bits $\rightarrow 2^{130}$ (NFS: $2^{80}$) ×
  - 2048 bits $\rightarrow 2^{130}$ (NFS: $2^{111}$) ×
  - 4096 bits $\rightarrow 2^{130}$ (NFS: $2^{149}$) ✔

Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
  - The Ideal-Cipher Model
  - The Generic Model
    - LD is hard
    - ECDSA
  - Shoup's Proof Technique
The Random-Permutation Model

Similar to the Random-Oracle Model, but with a permutation instead of a function

- It consists in considering some permutations as perfectly random permutations:
  - $\Lambda_p$ is initially set to an empty list
  - a query $x$ to $P$, or $y$ to $P^{-1}$, is answered by
    - if some $(x,y) \in \Lambda_p$, the corresponding value is returned
    - Otherwise, a random value ($y$ or $x$) is drawn from the appropriate range, $(x,y)$ is appended to $\Lambda_p$, $y$ or $x$ is returned

The Ideal-Cipher Model

An extension to the Random-Permutation model: A block cipher is seen as a family of truly random and independent permutations (for each key)

- The simulation works as follows:
  - $\Lambda_c$ is initially set to an empty list
  - a query $(k,m)$ to $E$, or $(k,c)$ to $D$, is answered by
    - if some $(k,m,c) \in \Lambda_c$, the corresponding value is returned
    - Otherwise, a random value ($c$ or $m$) is drawn from the appropriate range, $(k,m,c)$ is appended to $\Lambda_c$, $c$ or $m$ is returned
Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
  - The Ideal-Cipher Model
    - The Generic Model
      - LD is hard
      - ECDSA
  - Shoup's Proof Technique

The Generic Model

- It consists in considering the underlying group as a generic one: \( (G, +) \approx (\mathbb{Z}_q, +) \)
- But the adversary has access to the encoding of elements: \( E(Q) \)
- If one assumes that \( G = \langle P \rangle \), we define \( \sigma(x) = E(x \cdot P) \)
  \[ \sigma(x \pm y) = E((x \pm y) \cdot P) = E(x \cdot P \pm y \cdot P) \]
- Generic group: the encoding is random
Modeling a Generic Group

A usual way to model a generic group is to maintain a list $\Lambda$ which contains all $(x, \sigma(x))$:

- $\Lambda$ is initially set to $((1, \sigma(1)), (x, \sigma(x)))$: $P$ and $Y$
- A query $\sigma_1 \pm \sigma_2$ to the group law is answered:
  - First, because of the randomness of the encoding, there exist $x_1, x_2$, such that $(x_1, \sigma_1)$, $(x_2, \sigma_2) \in \Lambda$
  - If $(x_1 \pm x_2, \sigma) \in \Lambda$ for some $\sigma$, $\sigma$ is returned
  - Otherwise,
    - a new random $\sigma$ is drawn
    - $(x_1 \pm x_2, \sigma)$ is appended to $\Lambda$
    - $\sigma$ is returned

Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
  - The Ideal-Cipher Model
  - The Generic Model
  - LD is hard
  - ECDSA
- Shoup's Proof Technique
Proofs in the Generic Model

- $\Lambda \leftarrow ((P_1, \sigma_1), (P_2, \sigma_2))$:
  - $(P_1, \sigma_1) = (1, \sigma_1)$ encodes $P$, the generator
  - $(P_2, \sigma_2) = (X, \sigma_2)$ encodes $Y$, a random point ($Y = x.P$), thus $x$ is replaced by the unknown $X$

- For a query $\sigma_i \pm \sigma_j$, one looks for $(P_i, \sigma_i), (P_j, \sigma_j) \in \Lambda$
  - If $(P_i \pm P_j, \sigma) \in \Lambda$ for some $\sigma$, $\sigma$ is returned
  - Otherwise, $P_k = P_i \pm P_j$, a new $\sigma$ is drawn
    $(P_k, \sigma)$ is appended to $\Lambda$ and $\sigma$ is returned

Bad Simulation

- $\Lambda \leftarrow ((P_1 = 1, \sigma_1), (P_2 = X, \sigma_2))$: $(P, Y = x.P)$

- For a query $\sigma_i \pm \sigma_j$, one looks for $(P_i, \sigma_i), (P_j, \sigma_j) \in \Lambda$
  - If $(P_i \pm P_j, \sigma) \in \Lambda$ for some $\sigma$, it is returned
  - Otherwise, $P_k = P_i \pm P_j$, a new $\sigma$ is drawn

- A problem arises if $P \neq P'$ while $P(x) = P'(x)$
  But $Q = P - P' \neq 0$ is affine, then $\Pr[Q(x) = 0] \leq 1/q$
Bad Simulation (Cnt’d)

- After \( n \) queries to the group law oracle, at most \( n+2 \) polynomials have been defined.
- The probability that a problem arises is less than \( \frac{(n+1)(n+2)}{2q} \)

The Discrete Logarithm

- Let \( A \) be an adversary against the discrete logarithm: after \( n \) queries to the group law oracle, it outputs \( x \), the discrete logarithm of \( Y \) in the basis \( P \), with probability \( \varepsilon \)
- We replace the law oracle by the previous simulation: difference with probability less than \( \frac{(n+1)(n+2)}{2q} \)
- We do not need \( x \) in the simulation, it can be chosen at the very end:
  - Success probability less than \( \frac{1}{q} \)
The Discrete Logarithm (Cont'd)

- In the original game, the adversary outputs $z$, equal to $x$, with probability $\varepsilon$.
- In the final game, the adversary outputs $z$, equal to $x$, with probability less than $1/q$.
- Distance between the two games:
  \[(n+1)(n+2)/2q\]
- Conclusion:
  \[\varepsilon \leq 1/q + (n+1)(n+2)/2q \leq (n+2)^2/2q\]

Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
  - The Ideal-Cipher Model
  - The Generic Model
    - LD is hard
  - ECDSA
- Shoup's Proof Technique
Generic Model: ECDSA

\( G = \langle P \rangle \) and \( q \): common elements
\( x \): private key  \( Y = x.P \): public key

Signing \( m \):
- choose \( k \in \mathbb{Z}_q \) and compute \( R = k.P \)
- as well as \( r = f(R) \) and \( e = H(m) \)
- and \( s = (e+xr)/k \mod q \)

\[ \sigma = (r,s) \]

Verifying \((m,r,s)\):
- first \( 0 < r, s < q \)
- \( R' = e \ s^{-1} \cdot P + r \ s^{-1} \cdot Y \) and test if \( r = f(R') \)

Non-Malleability: ECDSA

Under some assumptions about the function \( f \)
and the hash function \( H \), one can show

- In the generic model,
one cannot break non-malleability of ECDSA
with probability significantly greater than

\[ (n+1)(n+q_s+1)/2q \]

- \( q_s \) is the number of signing queries
- \( n \) is the number of group law operations
Malleability: ECDSA

- In the description of ECDSA: $f(R) = x_R$ (the first coordinate of $R$)
- Thus $f(-R) = f(R)$
- If $(m,r,s)$ is a valid signature:
  $0 < r, s < q$ and $f(e s^{-1}.P + r s^{-1}.Y) = r$
- Then $(m,r,q-s)$ is a valid signature too:
  $s' = -s \mod q$ and $0 < r, s' < q$
  $f(e s'^{-1}.P + r s'^{-1}.Y) = f(-e s^{-1}.P - r s^{-1}.Y)$
  $= f(e s^{-1}.P + r s^{-1}.Y) = r$

Comments: ECDSA

- However, this function $f$ satisfies the requirements of the security theorem!
  Formal proof of a wrong fact
- The problem comes from the generic model
  - Indeed, when one knows $E(P)$, one knows $E(-P)$: they are not independent
  - Thus $f(R)$ and $f(-R)$ are not independent!
- If $f$ is assumed to behave like a random oracle
  - provably secure relative to DL in the random-oracle model only (KCDSA)
Summary

- Ideal Models
  - The Random-Oracle Model
    - The Forking Lemma
  - The Ideal-Cipher Model
  - The Generic Model
    - LD is hard
    - ECDSA

Shoup's Proof Technique

Reduction

Several problems $P_1$, $P_2$, $P_3$, ... may be reduced to a given attack $Atk$:

- Let $A$ be an adversary that breaks the scheme, with probability $\varepsilon$ on a probability space, defined by the internal coins, the external ones, the keys and the random oracles.
- We successively modify the probability space
Modifications

The modifications of the probability space $P$ may impact the success probability: $\varepsilon = \Pr[S]$

- Probability space $P$ unchanged
  $\Rightarrow$ the success probability is unchanged

- Probability space $P$ unchanged unless a bad event $E$ happens
  
  $S$ is independent of $E$: $\varepsilon' \geq \varepsilon \times \Pr[\neg E]$
  
  $\varepsilon' = \Pr[S'] = \Pr[S' \land E] + \Pr[S' \land \neg E]$
  
  $\geq \Pr[S' \land \neg E] = \Pr[S' \mid \neg E] \Pr[\neg E]$
  
  $\geq \Pr[S] \Pr[\neg E] = \varepsilon \times \Pr[\neg E]$

Shoup’s Lemma

The modifications of the probability space $P$ may impact the success probability: $\varepsilon = \Pr[S]$

- Probability space $P$ unchanged unless a bad event $E$ happens
  
  $S$ is not independent of $E$: $|\varepsilon' - \varepsilon| \leq \Pr[E]$

  $|\varepsilon' - \varepsilon| = |\Pr[S'] - \Pr[S]|$

  $= |\Pr[S' \land E] + \Pr[S' \land \neg E] - \Pr[S' \land E] - \Pr[S' \land \neg E]|$

  $= |\Pr[S' \mid E] \Pr[E] + \Pr[S' \mid \neg E] \Pr[\neg E] - \Pr[S \mid E] \Pr[E] - \Pr[S \mid \neg E] \Pr[\neg E]|$

  $= |\Pr[S' \mid E] - \Pr[S \mid E]| \times \Pr[E] \leq \Pr[E]$
Games

We thus define a sequence of games:

- Game 0 - the original attack - $\Pr[S_0] = \varepsilon$
- Game 1 - relation between $\Pr[S_0]$ and $\Pr[S_1]$
- Game $i$: which differs from Game $i-1$ only if event $E$ happens
  - Event $E$ happens = problem $P$ is broken
  - $\Rightarrow \Pr[E] \leq \text{Succ}^P(t')$
    where $t'$ is the running time of Game $i$
  - $\Rightarrow$ relation between $\Pr[S_{i-1}], \Pr[S_i], \text{Succ}^P(t')$

Final Game

- Game $n$: the success probability is easy to determine
  - typically: $\Pr[S_n] = 0$ or $\Pr[S_n] = \frac{1}{2}$
  - or also $\Pr[S_n] \leq q/2^k$

  $\Rightarrow$ one gets an upper-bound on $\varepsilon = \Pr[S_0]$

- Such a kind of proof clearly points out crucial points: the bad events to cancel
- The proof is easy to check/follow